DEPARTMENT OF MATHEMATICS

RAMLab User Guide

An Introduction to the Ryerson Applied Mathematics Laboratory

11/10/2010

Deparment of Mathematics
Ryerson University
350 Victoria Street
M5B 2K3

© 2010 by Department of Mathematics, Ryerson University, Toronto, ON, M5B 2K3

Version 1.0

Contents

List of Figures ..... **iii**
1 Introduction to the RAMLab **1**
1.1 Where is it and what equipment does it have? 1
1.2 How do I get access to the RAMLab? 1
1.3 How do I use this user guide? 1
1.3.1 Conventions used in this user guide 2
1.4 General safety guidelines 3
2 How to Log on to the RAMLab Locally **4**
3 Installing Cygwin (for Windows Users Only) **5**
3.1 Downloading Cygwin 5
3.2 Installing Cygwin 5
3.3 Running Cygwin for the first time 8
4 Logging on to the RAMLab Remotely **10**
4.1 Logging on to the RAMLab server from a Windows computer 10
4.2 Logging on to the RAMLab server from a Mac 10
4.3 How to log on to the RAMLab server from a Linux machine 11
4.4 Selecting a RAMLab workstation 12
4.5 Logging out of the RAMLab 13
5 Linux Basics **14**
5.1 The Linux file system 14
5.1.1 Finding what's in a directory: `ls` 14
5.1.2 Working with files and directories 17
5.2 Creating text files and executables with `pico` and `chmod` 19
5.3 Foreground and background processes 21
5.4 Using top and nohup with background processes 23
5.5 Ending a process with the `kill -9` command 26
6 Common Programs **28**
6.1 MATLAB 28
6.2 Intel FORTRAN (ifort) 28
6.3 The gcc c compiler 29
7 RAMLab Etiquette **30**

8 Quick Reference Guide 31

List of Figures

Figure 1 Selecting the openssh package for installation. 6
Figure 2 Selecting the xinit package for installation. 7
Figure 3 The output from pwd showing your home directory. 8
Figure 4 The output of the `usage` command and the factors you should consider when choosing a workstation. 12
Figure 5 The output from ls -l with some explanations. 15
Figure 6 The pico screen and some of the more common commands 20
Figure 7: The output from top and some of the more important information it gives you. 24

1 Introduction to the RAMLab

The Ryerson Applied Mathematics Laboratory (RAMLab) is a specialized computer facility for researchers in the Department of Mathematics at Ryerson University. The facility houses seven computer workstations that are equipped with software for running numerical computations and programs for a number of Applied Mathematics (AM) research fields. Researchers are usually graduate students in the Department of Mathematics but may also include undergraduate research assistants, professors, postdoctoral fellows, and visiting professors.

1.1 Where is it and what equipment does it have?

The RAMLab is located in room ENG239A of the George Vari Engineering and Computing Centre on the Ryerson University campus. The lab currently consists of seven Dell Dimension XPS 720 workstations. The operating systems installed on the computers are the 64 bit versions of Ubuntu Server 10.04 (5 workstations) and Windows (2 workstations). The numerical computation software packages include MATLAB and Intel FORTRAN.

1.2 How do I get access to the RAMLab?

Ask your professor or supervisor for permission to open a RAMLab account. Your professor will send a request to the Department of Mathematics' IT Specialist to provide you access. The IT Specialist will then contact you once your account has been created and give you your RAMLab user name and password. Your professor will guide you through the process of getting a key to ENG239A.

1.3 How do I use this user guide?

This user guide is a starting point for your use of the RAMLab. It will provide you with the following information:

- How to log on to the RAMLab from within Ryerson University.
- How to install the software you need to log on to RAMLab from outside Ryerson.
- How to log on to the RAMLab remotely.
- What the Linux operating system is and some of the basic commands you will need.
- How to run some of the more common Linux applications.
- What the RAMLab etiquette is.

1.3.1 Conventions used in this user guide

Command prompt, xterminal, and xterm: These terms are used interchangeably throughout the user guide.

The **command prompt** indicates the location where you will type commands and will often look something like this:

`*yourUserName*@epsilon:~$`

You will type commands after the `$`.

An **xterminal** is a window in which a command prompt is located (much like the Command Prompt application in Windows).

An **xterm** is short form for xterminal. It is also the command you type in the command prompt to bring up a new xterminal.

The `Courier` font: This font indicates either output from the workstation or a command that you will have to type into the command prompt. The following conventions are used:

1. `Regular courier font`: used to represent output from a workstation.
2. **`Bold courier font`**: used to represent commands that you will enter at the workstation **exactly as shown**. It is also used to indicate buttons to press (e.g. click on the **`Next`** button).
3. *`Italicized courier font`*: used to represent a value that you enter, but rather than entering the value exactly as shown, you will substitute the italicized text with your own value.

For example, if you see the following:

Type **`top -u`** *`yourUserName`* at the command prompt and press Enter.

you will know that you are to enter **`top -u`** exactly as shown, and replace *`yourUserName`* with your own value (in this case your Department of Mathematics user name).

Your user name: Unless otherwise stated, when you are asked to input your user name (e.g. *`yourUserName`*) this refers to your Department of Mathematics user name. This is the user name you will be given by the IT Specialist.

Entering commands: After typing text into a command prompt make sure to press the Enter key. For example, if you see the following:

Enter the command `ls -l` in the command prompt.

You will know that you are to type the command `ls -l` exactly as shown into the command prompt and then press Enter.

1.4 General safety guidelines

- Keep the area around the workstations free of clutter and liquids.
- Do not attempt to modify, upgrade or troubleshoot the workstations. If you need to perform any of these tasks, contact the IT Specialist.
- For emergency assistance dial 80 on the telephone provided in the RAMLab.

2 How to Log on to the RAMLab Locally

Go to the RAMLab in ENG239A, located in the George Vari Engineering and Computing Centre. Choose any terminal that is free. If the terminal is locked, press the **`Switch User`** button on the display. Enter your Department of Mathematics user name and password. If you have any problems logging in, contact the IT Specialist for the Department of Mathematics.

If you want to log on to the lab from outside of ENG239A, follow the instructions in section 4 *Logging on to the RAMLab Remotely* of this user guide. If you want to log on to the lab from home using a Windows computer, you must first install Cygwin. See the instructions in section 3 *Installing Cygwin (for Windows Users Only)* before attempting the instructions in section 4.1 *Logging on to the RAMLab server from a Windows computer*.

3 Installing Cygwin (for Windows Users Only)

Cygwin is a Linux-like environment that you can install in Windows. In order to connect to the RAMLab you need an xterminal (xterm) and an X Windows server installed on your Windows machine. Cygwin provides both. This section walks you through downloading, installing, and running Cygwin. In order to complete the installation you must be connected to the internet and be an administrator on your computer.

3.1 Downloading Cygwin

1. In Windows, open your internet browser and go to http://x.cygwin.com/.
2. Scroll down the page to the "Downloading and Installing" section and click on the setup.exe link or on any of the "Install Cygwin/X Now" icons on the right hand side of the page.

   Result: A window that asks if you would like to save the setup.exe file pops up.
3. Click on **`Save file`** to download the setup file on your computer.

   Additional information: A window may pop up that asks if you want to run the downloaded file. If you want to install Cygwin right now, press the **`Run`** button and continue on to step 2 of 3.2 *Installing Cygwin*.

3.2 Installing Cygwin

1. Go to the location where the setup.exe file was saved (usually the Downloads folder in your Windows home directory) and run the file by double-clicking on it.

   Additional information: If the Windows User Account Control prompt pops up, click on the **`Allow`** option to install the program. After this, the Cygwin Net Release Setup Program window pops up. Press the **`Next`** button to start downloading and installing Cygwin.
2. In the next window, select the **`Install from Internet`** option and press the **`Next`** button.

   Additional information: It is a good idea to keep the downloaded files because you may need to install more features later.
3. Keep the default choices for the root directory and allowing other users to use Cygwin and click on the **`Next`** button.

   Note: If you do not install Cygwin in the default folder (c:\cygwin) then make a note of where it is installed so you will know where to find it later.
4. Select where you want to download the Cygwin files (it is fine to use the default choice) and then click the **`Next`** button.
5. Select your internet connection type. If you are not sure which type to choose, leave it as the default choice, **`Direct Connection`**. After you have chosen your connection press the **`Next`** button.

6. The next window will ask you to select a download site. It is best to choose the nearest location to you (http://mirror.csclub.uwaterloo.ca). After you have chosen the download site press the **Next** button.

   Result: A window with a progress bar pops up. If the download stalls, you can stop it by clicking the **Cancel** button. Go back to step 1 of this section and try a different location.

   Additional information: After Cygwin has finished downloading, a Setup Alert window pops up that warns you about the possible conflicts that a new installation of Cygwin may have with previous installations. If this is your first time installing Cygwin, it is safe to click **OK** to continue. If you have other installations of Cygwin, you may want to visit http://cygwin.com to see what issues you may run into by installing this version.

7. You should now see the Select Packages window. You need to install two packages: openssh and xinit. We will first install openssh:

   7.1. Enter **ssh** in the Search text box.

   7.2. Click on the + icon next to Net to expand the packages in Net.

   7.3. Notice the package called openssh at the bottom of the list. It is currently not set to be installed (Cygwin is set to skip this package). Click on the icon.

   Result: The word Skip will be replaced by a version number, which indicates that openssh will be installed. Your window should look like Figure 1 (the version number may be different).

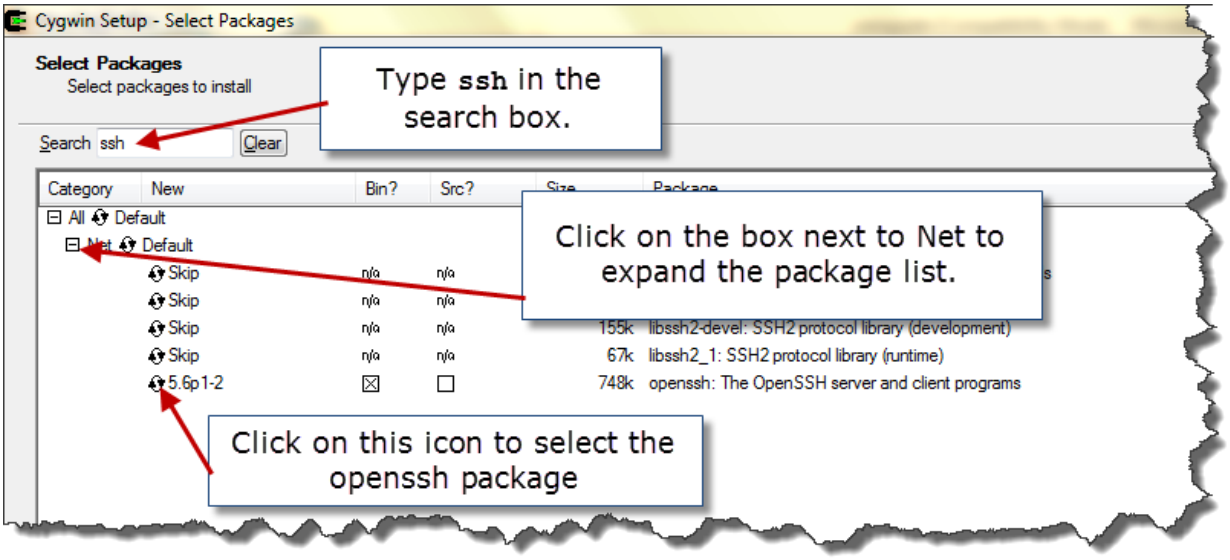

Figure 1 Selecting the openssh package for installation.

8. We will now install xinit:

    8.1. Enter **`xinit`** in the Search text box.

    8.2. Click on the + icon next to X11 to expand the packages in X11.

    8.3. Select the xinit package by clicking on the icon. When xinit has been properly selected your window should look like Figure 2 (the version number may be different).

    8.4. Press the **`Next`** button to continue with the installation.

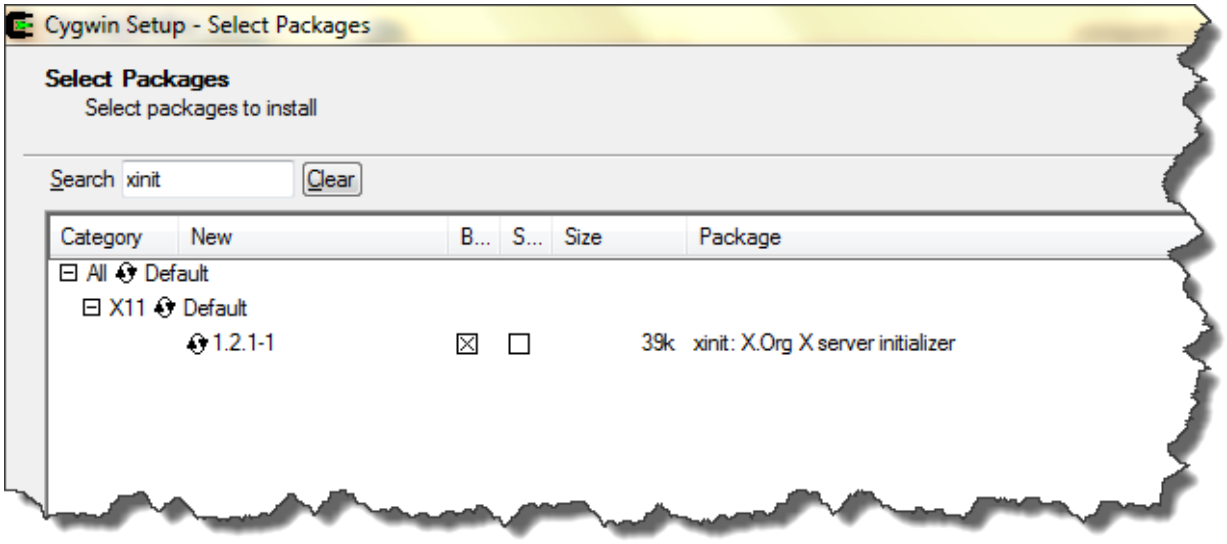

Figure 2 Selecting the xinit package for installation.

9. The next window is the Resolving Dependencies window. Make sure that the box next to Select Recommended Packages (RECOMMENDED) has a check mark in it and then press the **`Next`** button.

    Result: A window showing the installation progress pops up. It can take a few minutes to an hour to download and complete the installation depending on your internet connection.

10. Once the installation is complete, a final window asks if you would like to install a desktop icon and a Start menu icon for Cygwin. It is a good idea to say yes to both of these. Press the **`Finish`** button to complete the setup of Cygwin.

    Note: If a window pops up asking you whether the program installed, verify that the program installed correctly.

3.3 Running Cygwin for the first time

This section shows you how to start Cygwin, locate your home directory in Cygwin, and create a shortcut from your Windows desktop to your Cygwin home directory.

1. On your computer go to **`Start->All Programs->Cygwin-X`** and click on the program called **`XWin Server`**.

   Result: An xterminal pops up. This allows you to run graphical programs on the remote servers. If a window pops up asking you whether or not to unblock the program, press the **`Unblock`** button.

2. Type **`cd`** in the xterminal and press Enter.

   Additional information: Typing the **`cd`** command forces you into your home directory.

3. Type the command **`pwd`** in the xterminal and press Enter.

   Result: The command **`pwd`** stands for "**p**rint **w**orking **d**irectory" and shows you the directory you are currently in. You should get an output like the one in Figure 3, except that your output will look like `/home/yourWindowsUsername`.

   We will now create a link to this home directory on your Windows desktop so that you can easily send files back and forth between Windows and your Cygwin environment.

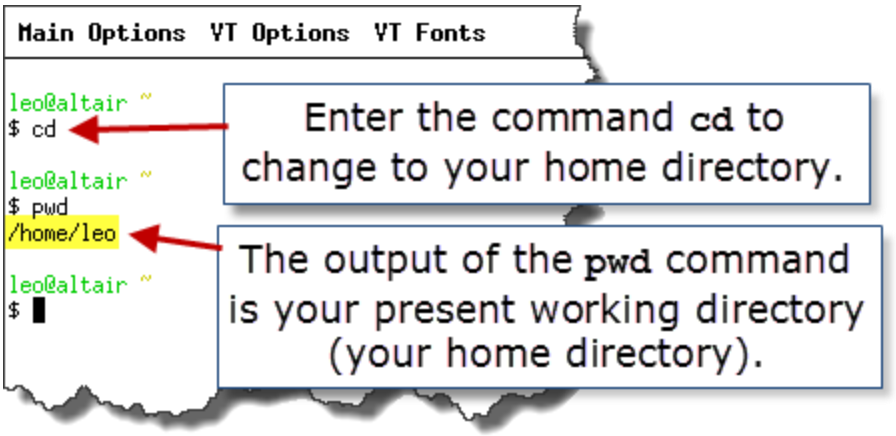
Main Options VT Options VT Fonts
leo@altair ~
$ cd
leo@altair ~
$ pwd
/home/leo
leo@altair ~
$
Enter the command cd to change to your home directory.
The output of the pwd command is your present working directory (your home directory).

Figure 3 The output from pwd showing your home directory.

4. Move your mouse pointer to an empty area of your desktop. Right-click on the desktop and from the drop-down menu select **`New->Shortcut`**.

5. Type the following in the text box:

   `c:\cygwin\home\`*`yourWindowsUsername`*

   Note: *`yourWindowsUsername`* is part of the output from step 3.

   Additional information: At this point it is assumed that you have installed Cygwin in the default location, which is c:\cygwin on your hard drive. If you

selected a different folder during installation, replace **`c:\cygwin`** with the folder you selected.

6. Press the **`Next`** button. Change the name of the shortcut if you prefer (we would suggest something like cygwinHome) and press the **`Finish`** button. The shortcut should be on your desktop.

   Result: Cygwin is now installed on your computer. You can now connect to the RAMLab remotely from your Windows computer (see section 4.1 *Logging on to the RAMLab server from a Windows computer*).

4 Logging on to the RAMLab Remotely

The RAMLab is located on a private internal network, which means that you cannot connect to it directly from outside Ryerson. In order to connect to the RAMLab remotely, you must first log on to a gateway server called the RAMLab Server (ramlab.math.ryerson.ca). How you do this will differ depending on which operating system you use.

4.1 Logging on to the RAMLab server from a Windows computer

It is assumed that you have already installed Cygwin with the ssh and xinit packages. If this is not the case please follow the instructions in section 3 *Installing Cygwin (for Windows Users Only)*.

1. If you have not already done so, open a Cygwin xterm by going to **Start->All Programs->Cygwin-X->XWin Server**.
2. Type **ssh –X** *yourUserName*@**ramlab.math.ryerson.ca** in the command prompt and press Enter.

   Additional information: The user name you enter is your RAMLab user name, given to you by the IT Specialist for the Department of Mathematics.
3. Type your password and press Enter.

   Note: As you type your password the command prompt stays blank. This is normal.

   Additional information: The first time you log on to the RAMLab server you are asked if you want to connect to this server. Type **yes** and press Enter. If you have trouble logging on, make sure that you entered the correct user name in front of **ramlab.math.ryerson.ca** and that you do not have Caps Lock on. If you still have problems connecting, speak with the math department's IT Specialist.

If you logged in correctly, you should see the following as a command prompt:

yourUserName@**RAM-lab-server:~$**

Your next step is to find a free workstation to use (see *4.4 Selecting a RAMLab workstation*).

4.2 Logging on to the RAMLab server from a Mac

1. Open an xterminal window by opening Finder, going into Applications (found on the left menu), opening the Utilities folder and double clicking Terminal.

   Optional: You can also open an xterminal window by opening Spotlight (hold down the **Command** key and press the **space bar**) and entering **terminal** in the Spotlight search box. Press Enter and select the **terminal** program.
2. Type **ssh –X** *yourUserName*@**ramlab.math.ryerson.ca** in the command prompt and press Enter.

Additional information: The user name you enter is your RAMLab user name, given to you by the IT Specialist for the Department of Mathematics.

3. Type your password and press Enter.

   Note: As you type your password the command prompt stays blank. This is normal.

   Additional information: The first time you log on to the RAMLab server you are asked if you want to connect to this server. Type **`yes`** and press Enter. If you have trouble logging on, make sure that you entered the correct user name in front of **`ramlab.math.ryerson.ca`** and that you do not have Caps Lock on. If you still have problems connecting, speak with the math department's IT Specialist.

If you logged in correctly, you should see the following as a command prompt:

*`yourUserName`***`@RAM-lab-server:~$`**

Your next step is to find a free workstation to use (see section *4.4 Selecting a RAMLab workstation*).

4.3 How to log on to the RAMLab server from a Linux machine

1. Open an xterminal by holding down **`Alt`** and pressing **`F2`** to bring up the Run Application window. Type **`xterm`** in the command entry box and press Enter.

   Additional info: Depending on your distribution and window manager you may want to use a different terminal program (e.g. gnome-terminal).

2. Type **`ssh -X`** *`yourUserName`***`@ramlab.math.ryerson.ca`** in the command prompt and press Enter.

   Additional information: The user name you enter is your RAMLab user name, given to you by the IT Specialist for the Department of Mathematics.

3. Type your password and press Enter.

   Note: As you type your password the command prompt stays blank. This is normal.

   Additional information: The first time you log on to the RAMLab server you are asked if you want to connect to this server. Type **`yes`** and press Enter. If you have trouble logging on, make sure that you entered the correct user name in front of **`ramlab.math.ryerson.ca`** and that you do not have Caps Lock on. If you still have problems connecting, speak with the math department's IT Specialist.

If you logged in correctly, you should see the following as a command prompt:

*`yourUserName`***`@RAM-lab-server:~$`**

Your next step is to find a free workstation to use (see *4.4 Selecting a RAMLab workstation*).

4.4 Selecting a RAMLab workstation

Once you have logged on to the RAMLab you must select an available workstation. You can select any one of the Linux configured workstations (i.e. Alpha, Beta, Delta, Epsilon, or Zeta), but we recommend that you select a workstation that is not in heavy use.

1. Log on to the RAMLab server (see section 4 *Logging on to the RAMLab Remotely*).
2. Type the command **`usage`** in the xterm and press Enter.

   Result: You should get an output similar to what is shown in Figure 4 (you may have to enlarge the xterm window to see all of the output).The **`usage`** command displays the current state of each of the Linux workstations. The important numbers to look at are the system load (the closer to zero the better), the number of processes running (the lower the better), and the memory usage (the closer to 0% the better). Some other factors you may want to consider are the number of users logged on to the machine and the usage of /home (especially if you plan on generating large output files).

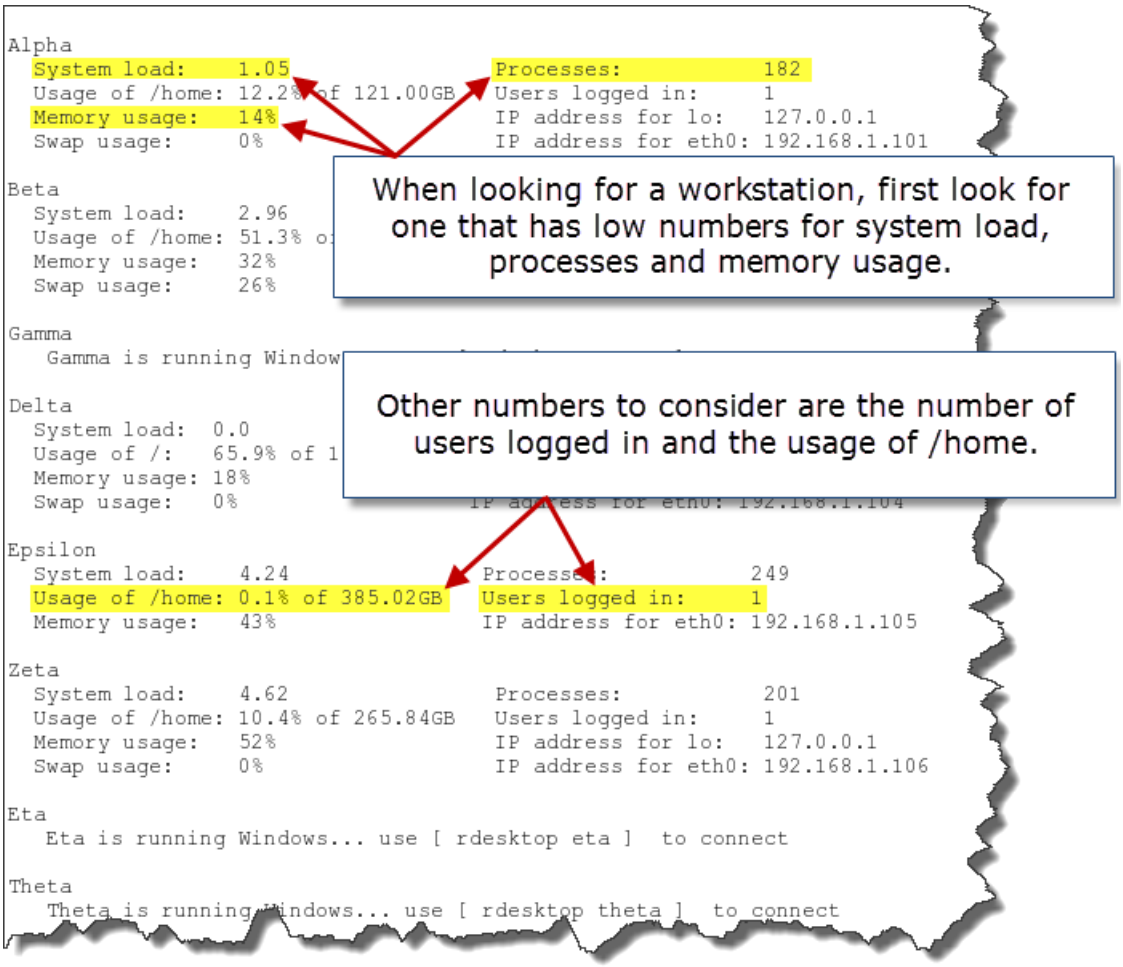
Alpha
System load: 1.05
Usage of /home: 12.2% of 121.00GB
Memory usage: 14%
Swap usage: 0%
Processes: 182
Users logged in: 1
IP address for lo: 127.0.0.1
IP address for eth0: 192.168.1.101
Beta
System load: 2.96
Usage of /home: 51.3% o
Memory usage: 32%
Swap usage: 26%
Gamma
Gamma is running Window
Delta
System load: 0.0
Usage of /: 65.9% of 1
Memory usage: 18%
Swap usage: 0%
Epsilon
System load: 4.24
Usage of /home: 0.1% of 385.02GB
Memory usage: 43%
Processes: 249
Users logged in: 1
IP address for eth0: 192.168.1.105
Zeta
System load: 4.62
Usage of /home: 10.4% of 265.84GB
Memory usage: 52%
Swap usage: 0%
Processes: 201
Users logged in: 1
IP address for lo: 127.0.0.1
IP address for eth0: 192.168.1.106
Eta
Eta is running Windows... use [rdesktop eta] to connect
Theta
Theta is running Windows... use [rdesktop theta] to connect
When looking for a workstation, first look for one that has low numbers for system load, processes and memory usage.
Other numbers to consider are the number of users logged in and the usage of /home.

Figure 4 The output of the `usage` command and the factors you should consider when choosing a workstation.

3. Once you have selected the workstation you want to work on, type the name of the workstation in the xterm in lower case letters and press Enter.

   Result: You should see several messages telling you that you are connected to the workstation. Also, your command prompt should look something like:

   yourUserName`@`*workstation*`:~$`

   Additional information: The possible choices for workstations are **`alpha`**, **`beta`**, **`delta`**, **`epsilon`**, and **`zeta`**. Please note that your Work directory will move to whatever workstation you choose, so you will always have access to this directory. Therefore, you do not need to log on to the same workstation every time to access your files. Simply make sure that your files are in your Work directory (you will learn about moving files in section 5.1.2 *Working with files and directories*) and you can choose whichever workstation best meets your needs at the time.

4.5 Logging out of the RAMLab

1. Type **`exit`** in the xterm and press Enter.

   Result: You return to the RAMLab sever command prompt.

2. Type **`exit`** in the xterm and press Enter.

   Result: You are logged out of the RAMLab.

5 Linux Basics

In order to make the best use of the RAMLab you must learn how to use the Linux operating system. This section introduces you to the basics of Linux by providing short tutorials on the Linux file system and some of the more common Linux commands and tasks. This is only an introduction to get you started using the system. For more information, we suggest consulting *Unix Shell Programming* by Lowell Jay Arthur and Ted Burns.

If you are already familiar with either the Linux or UNIX operating systems, then feel free to skim over this section or skip it entirely. If you have never used these operating systems, then please go over this section and the tutorials as they will provide you with the background you need to start using the RAMLab environment.

5.1 The Linux file system

The Linux file system has a directory tree structure. The base of the directory tree is called the root directory and is represented by **/** (forward slash). All other directories in the system spread from the root directory like branches on a tree. The next two sections are tutorials that will familiarize you with the Linux file system.

5.1.1 Finding what's in a directory: `ls`

To see what files are in your current directory, enter the command **`ls`** (**`ls`** is short form for **l**i**s**ting). For example, if you enter **`ls`** right after logging on to a workstation, you will get an output that looks like this:

```
documents  output01  run01
```

This is a list of what is in your current directory, but at this point it is not known which of these items is a file and which is a directory. To get more information about these items, use the **`-l`** option (this is the **letter** **`l`** on the keyboard, **not** the number one) with **`ls`** to get a detailed listing:

Type the command **`ls -l`** and press Enter.

This gives you a detailed listing with more information about the items. Figure 5 depicts results from this command and some important features.

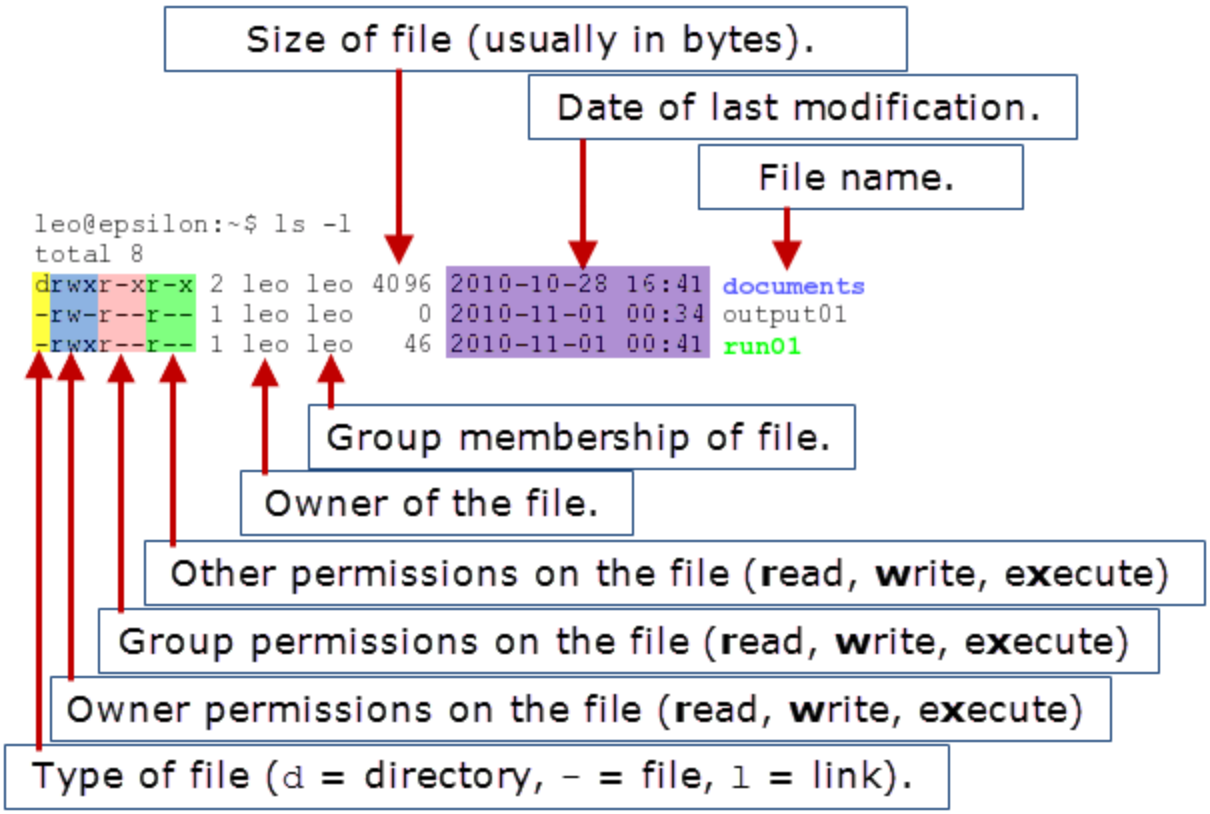
Size of file (usually in bytes).
Date of last modification.
File name.
leo@epsilon:~$ ls -l
total 8
drwxr-xr-x 2 leo leo 4096 2010-10-28 16:41 documents
-rw-r--r-- 1 leo leo 0 2010-11-01 00:34 output01
-rwxr--r-- 1 leo leo 46 2010-11-01 00:41 run01
Group membership of file.
Owner of the file.
Other permissions on the file (read, write, execute)
Group permissions on the file (read, write, execute)
Owner permissions on the file (read, write, execute)
Type of file (d = directory, - = file, l = link).

Figure 5 The output from ls -l with some explanations.

The most important information is in the first column, which is made up of ten characters. This column tells you what kind of file you have, as well as the permissions on the file.

The first character in the first column tells you the type of file you have:

- **`d`**: directory
- **`-`**: file
- **`l`**: link to another file (similar to a shortcut in Windows).

The next three characters in the first column are the permissions that the owner has on the file or directory (the owner is given in the third column of the output and is usually your user name):

- **`r`**: The owner can read the contents of the file. If it is a directory then the owner can run **`ls`** in the directory to see a listing of what it contains.
- **`w`**: The owner can change the contents of the file. If it is a directory then the owner can create and delete files in the directory.
- **`x`**: The owner can run the file. If it is a directory then the owner can move into the directory using a command called **`cd`**.

The next three characters in the first column are the permissions for members of the same group as the file (the group is given in the fourth column of your output).

The final three characters in the first column are the permissions for anyone else on the system.

It is important to know how to check the permissions on a file or directory because most of the time, problems in Linux are related to file permissions that have been set incorrectly.

Always start troubleshooting by looking at the file and directory permissions.

The **`ls -l`** command displayed a detailed listing of the contents of the directory, but it did not list all of the files in the home directory. To get a listing of all of these files, use the **`–al`** option with the **`ls`** command:

Type **`ls –al`** at the command prompt in your home directory and press Enter.

You should get an output similar to the following:

```
leo@epsilon:~$ ls -al
total 84
drwxr-xr-x 14 leo  leo  4096 2010-11-01 10:13 .
drwxr-xr-x  9 root root 4096 2010-10-21 16:06 ..
-rw-------  1 leo  leo   610 2010-11-01 01:28 .bash_history
-rw-r--r--  1 leo  leo   220 2010-10-14 21:14 .bash_logout
-rw-r--r--  1 leo  leo  3103 2010-10-14 21:14 .bashrc
drwx------  2 leo  leo  4096 2010-10-14 21:18 .cache
drwxr-xr-x  2 leo  leo  4096 2010-10-14 22:23 .config
drwx------  3 leo  leo  4096 2010-10-14 22:23 .dbus
drwxr-xr-x  2 leo  leo  4096 2010-10-28 16:41 documents
drwx------  3 leo  leo  4096 2010-10-28 16:56 .gconf
drwx------  2 leo  leo  4096 2010-10-28 16:57 .gconfd
drwxr-xr-x  3 leo  leo  4096 2010-10-28 16:56 .gnome2
-rw-r--r--  1 leo  leo     0 2010-11-01 00:34 output01
-rw-r--r--  1 leo  leo   675 2010-10-14 21:14 .profile
-rw-------  1 leo  leo   218 2010-10-28 16:56 .recently-used.xbel
-rwxr--r--  1 leo  leo    46 2010-11-01 00:41 run01
```

In this listing the previous files are still here (documents, output01, and run01) but there are now files that have a '.' as the first character in the file name. In Linux, any file that has a period as the first character in its name is hidden and can only be seen if you use the **`–al`** option with **`ls`** (**`-al`** tells **`ls`** to display all of the files in the directory).

Notice that the first two directories listed are named '.' and '..'. These are very important directories:

- '.' is how Linux refers to the directory you are currently in.
- '..' is how Linux refers to the directory one level up from your current directory (the parent of your current directory).

For example, if you are in your home directory (`/home/yourUserName`), '.' refers to `/home/yourUserName` and '..' refers to `/home`.

Quick Quiz:

You should be able to answer the following questions about finding files in a directory:

A. How do you list all files in a directory?
B. What permissions do you have in your directory when you see the following:

```
drwx------
```

C. How does Linux refer to your current directory?
D. How does Linux refer to the parent of your current directory?

5.1.2 Working with files and directories

A directory is a good location to organize your files. This section provides a small tutorial on how to create a directory, move into that directory, and move files to that directory. To do this, use the following commands:

`touch`: create an empty file

`cd`: change to another directory (**c**hange **d**irectory)

`mkdir`: create a directory (**m**a**k**e **dir**ectory)

`pwd`: print out current working directory (**p**rint **w**orking **d**irectory)

`cp`: copy files or directories from one location to another (**c**o**p**y)

`mv`: move files or directories from one location to another (**m**o**v**e)

`rm`: remove or delete a file permanently from the computer (**r**e**m**ove)

1. Log on to one of the workstations and have an xterm available for this tutorial (see section 2 *How to Log on to the RAMLab Locally* or section 4 *Logging on to the RAMLab Remotely*). Type the command **`cd`** in the xterm and press Enter.

   Additional information: The **`cd`** command takes you to your home directory. The ~ in the resulting command prompt indicates that you are in your home directory.

2. Type the command **`touch myFile`** at the command prompt and press Enter. To see the results of this action, type the command **`ls -l`** and press Enter.

   Result: You should see a file called `myFile` in the output. The command **`touch myFile`** creates an empty file with the name `myFile`. The output from the **`ls -l`** command verifies this. Notice the permissions on the file that you created. You should be the owner of the file and be able to read it and write to it. You should also see a directory called Work in the listing. If you do not have a Work directory, let the IT Specialist know.

We will now create a directory into which we will move the file.

3. Type the command **`cd Work`** in the xterm and press Enter.

   Result: You are now in your Work directory. As mentioned previously, your Work directory is available on any workstation you use in the RAMLab. You should keep all of your files in this directory.

   Additional information: If the directory you want to change to is inside your current directory, then you only need to type the name of the directory after **`cd`**. Otherwise, you will need to give **`cd`** the full path to the directory (which in this case would be `/home/`*`yourUserName`*`/Work`).

4. Type the command **`mkdir myDirectory`** at the command prompt and press Enter. To see the results of this action, type the command **`ls –l`** and press Enter.

   Result: There is a new directory called `myDirectory`. Notice that you are the owner and that you have read, write and execute permissions on the directory.

5. Type the command **`cd myDirectory`** and press Enter.

   Result: Your current working directory should now be `myDirectory`. You will verify this in the next step.

6. Type the command **`pwd`** and press Enter.

   Result: The **`pwd`** command lets you see what directory you are currently in (in this case **`myDirectory`**). The output from this command will be something like:

   `/home/`*`yourUserName`*`/Work/myDirectory`.

   We will now transfer files to this directory.

7. Type **`cd ..`** and press Enter to go back to your `Work` directory. Type **`cd ..`** and press Enter to go back to your home directory.

   Note: There **must** be a space between **`cd`** and **`..`** when you type this command or you will get an error.

   Additional information: When you enter the command **`cd ..`** you are telling Linux to change directories to the parent of the directory that you are currently in.

8. Type the command **`cp myFile Work/myDirectory/myFileCopy`** at the command prompt and press Enter. To see the results of this action, type the command **`ls –l`** and press Enter.

   Result: Your original file, `myFile`, is still in your home directory, and there is a copy of `myFile` called `myFileCopy` in `Work/myDirectory`. You will verify this in the next step.

9. Type the command **`mv myFile Work/myDirectory/myFileBackup`** at the command prompt and press Enter. To see the results of this action, type the command **`ls -l`** and press Enter.

   Result: The file `myFile` is no longer in your home directory. The **`mv`** command moves a file to the destination you give it (in this case to a file called `myFileBackup` in your `Work/myDirectory` directory) and then deletes the original file. In the next step you will verify that the file was indeed moved.

10. Type the command **`cd Work/myDirectory`** at the command prompt and press Enter. Type the command **`ls -l`** and press Enter to see all the files in the directory.

    Result: You should see the two files you put in `Work/myDirectory`: `myFileCopy` and `myFileBackup`.

    Note: If you want to delete the files you just created, type the command **`rm myFileCopy myFileBackup`** at the command prompt and press Enter. Type the command **`ls -l`** and press Enter to see the result. The files should be gone.

CAUTION: **Files that are deleted using `rm` cannot be recovered! Be very careful when using rm!**

Quick Quiz:

You should be able to answer the following questions about working with files and directories in Linux:

A. How do you change from one directory to another?
B. How do you create a new directory?
C. How do you copy files from one directory to another?
D. How do you move a file?
E. How do you remove a file permanently from a directory?

5.2 Creating text files and executables with `pico` and `chmod`

This tutorial shows you how to create a file using **`pico`**, one of the text editors that is available on the Linux workstations. The **`pico`** text editor is a very simple program which we recommend using (especially if you are logging in from a remote connection). **`Pico`** uses very little resources, is intuitive, and will meet most of your text editing needs. This tutorial also shows you how to make the file you create in **`pico`** an executable Linux script.

1. Log on to the RAMLab (see section 2 *How to Log on to the RAMLab Locally* or 4 *Logging on to the RAMLab Remotely*).

2. Type **`pico myExecutable`** in the xterm and press Enter.

   Result: An empty pico text editing window pops up. Notice that there is a cursor at the top left corner of the screen. Along the bottom is a menu with some of the most common commands. The **^** stands for the control (ctrl) key on your keyboard. To perform a command, hold down the ctrl key and press the letter that corresponds to the desired command. For example, to Write Out

(pico's way of saying "to save") your file, hold down the ctrl key and press the 'o' on your keyboard. Figure 6 shows the commands for cutting, pasting, saving, and exiting in pico.

3. Enter the following lines of text in the pico window (make sure to press Enter after each line):

```
#!/bin/bash
sleep 30
echo 'myExecutable has finished'
```

Result: Your window should look like the one in Figure 6. Save the file (press ctrl+o and then press Enter to save the file as myExecutable). Exit pico (ctrl+x).

Additional information: The file myExecutable is a Linux script: a file that contains a number of Linux commands to be executed. Our script has three commands: the **`#!/bin/bash`** command tells Linux to run the script in a bash shell, the **`sleep 30`** command pauses the program for 30 seconds, and the **`echo 'myExecutable has finished'`** command will display the line `myExecutable has finished` in the xterm.

To run the file, type its name (preceded by its directory) and press Enter. Since you are in the same directory as the file you can simply type '`./`' as the directory for the file (you are telling Linux that the file is contained in the current directory) and follow that with the name of the file to run. You will do this in the next step.

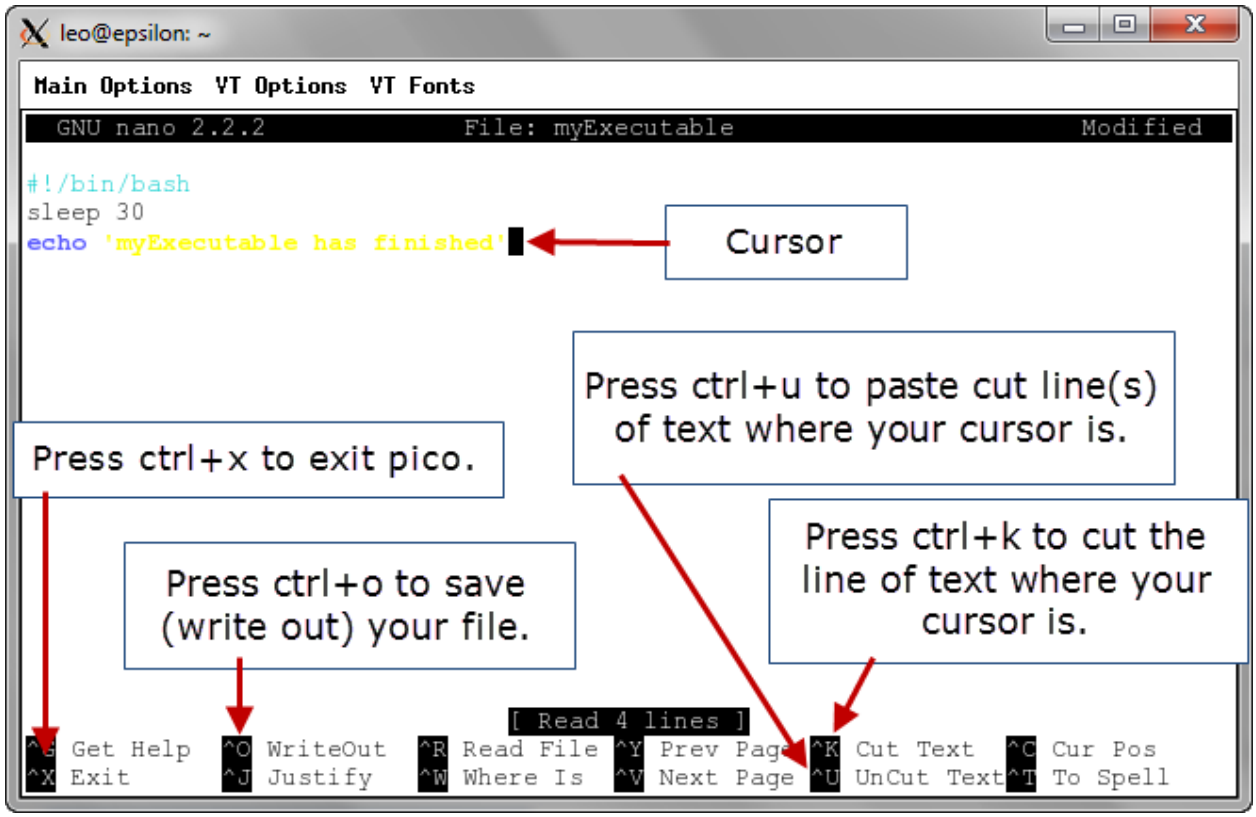
leo@epsilon: ~
Main Options VT Options VT Fonts
GNU nano 2.2.2 File: myExecutable Modified
#!/bin/bash
sleep 30
echo 'myExecutable has finished'
Cursor
Press ctrl+u to paste cut line(s) of text where your cursor is.
Press ctrl+x to exit pico.
Press ctrl+o to save (write out) your file.
Press ctrl+k to cut the line of text where your cursor is.
[Read 4 lines]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Figure 6 The pico screen and some of the more common commands

4. Type the command **`./myExecutable`** in the xterm and press Enter.

   Result: You should get an output similar to the following:

   ```
   -bash: ./myExecutable: Permission denied
   ```

5. Type **`ls -l`** in the xterm and press Enter.

   Additional information: The output from this command shows that you own the file and that you have read and write permissions, but you do not have execute permission on the file. Although the file contains executable commands, the file itself is not executable. In order for Linux to run the file you have to make it executable. You will do this in the next step.

6. Type the command **`chmod u+x myExecutable`** at the xterm and press Enter. Type **`ls -l`** in the xterm and press Enter**.**

   Additional information: The command **`chmod`** stands for **ch**ange **mod**e and is used to set the permissions on a file. In this case we are using it to add execute permissions for the user that owns the file (this is done with the **`u+x`** flag). The output of the **`ls -l`** command will verify that you now have execute permission on the file.

7. Type **`./myExecutable`** in the xterm and press Enter.

   Result: The xterm hangs for 30 seconds and then outputs `myExecutable has finished`.

This tutorial has shown you how to use the text editor **`pico`** to create files. You also created an executable file called myExecutable that contained Linux commands. These commands are processed when the script is executed (or called). You also saw that by default, files that you create do not have execute permissions set, but you can use the **`chmod`** command to set the execute permissions for the file. The next tutorial will discuss executables in more detail, especially how to manage them.

Quick Quiz:

You should be able to answer the following questions about creating files in Linux:

A. What is pico?
B. In pico, how do you save a file? How do you exit pico?
C. What symbol in the pico window stands for the `ctrl` key on the keyboard?
D. How do you run an executable?
E. What does the command **`chmod u+x myExecutable`** do?

You will be using the myExecutable file for some of the next tutorials, so for now do not delete it.

5.3 Foreground and background processes

In the previous section we created an executable file that waited 30 seconds and then finished. You probably noticed that while the executable (or program) was running, it took control of the xterm and you had to wait for the process to end

before regaining control of the xterm. In Linux this is called running a process in the foreground. For short programs this is not a big problem, but what if you had a program that took an hour to run? It would be more ideal if you could start it and work on something else at the same time, or even log off and come back later to see the results. Linux allows you to do this by starting programs in the background.

What follows is a short tutorial on how to start a program in the foreground and then how to start the program in the background. For this tutorial you will need the myExecutable file that you wrote in section *5.2 Creating text files and executables with `pico` and `chmod`*. If you have not completed that section please do so before attempting this tutorial.

1. Log on to the RAMLab (see section 2 *How to Log on to the RAMLab Locally* or 4 *Logging on to the RAMLab Remotely*).
2. Go to the directory where you stored the myExecutable program that you created in section *5.2 Creating text files and executables with `pico` and `chmod`*.
3. Type the command **`./myExecutable`** in the xterm and press Enter.

   Result: The program will pause for 30 seconds and then output a line that says `myExecutable has finished`. Notice that while the program was running, you could not interact with the xterm.
4. Type **`./myExecutable &`** in the xterm and press Enter.

   Additional information: The **`&`** at the end of the command tells Linux that you want to run the command in the background. This means that the program will run in another part of operating system and you will be able to work on other tasks while the program is executing. You may get an output similar to the following:

   ```
   [1] 5010
   ```

   The number in the square brackets is the job number and the second number is the process identification number (or pid) for the process.
5. Wait about 30 seconds and then press Enter in the xterm.

   Result: After about 30 seconds the output from./myExecutable, `myExecutable has finished`, was displayed and after you pressed Enter the following output was displayed:

   ```
   [1]+  Done                    ./myExecutable
   ```

   This tells you that the process you were running, which was called `./myExecutable` and had job number 1, is done.

You might be wondering what happens to your background processes if you log out. This is covered in the next tutorial.

Quick Quiz:

You should be able to answer the following questions about running a process in the background:

A. What is the difference between a foreground and a background process?
B. What kinds of processes would you want to run in the background?
C. How do you start a background process?
D. How do you know that a background process is finished?

5.4 Using top and nohup with background processes

Section 5.3 *Foreground and background processes* showed you how to start a process in the background. This is a good option for programs that meet the following criteria:

- The program does not require any input from you.
- The program will take a long time to finish executing.

The second point is especially important if you want to work on something else at the same time or if you have to log out of the system before the program has finished executing.

What follows is a short tutorial that shows you how to track the status of a background process and its output. For this tutorial you will need the myExecutable file that you wrote in section *5.2 Creating text files and executables with `pico` and `chmod`*. If you have not completed that section please do so now and then return to this tutorial.

1. Log on to the RAMLab (see section 2 *How to Log on to the RAMLab Locally* or 4 *Logging on to the RAMLab Remotely*).

2. Type the command **`xterm &`** and press Enter.

   Result: Another xterm pops up that is also logged on to your current workstation. For the rest of this tutorial we will call the first xterm the original xterm (this is the one you typed the **`xterm &`** command into) and the new xterminal the second xterm.

   Additional information: It may take a few minutes for the second xterm to appear.

3. Type the command **`top -u`** *`yourUserName`* in the original xterm and press Enter. Keep this window open for the rest of the tutorial.

   Result: You should see something similar to Figure 7, except that *`youUserName`* will appear instead of `leo`. The output from **`top -u`** *`yourUserName`* shows you all the commands that you are currently running and sorts them according to what percentage of the CPU you are using. The list is updated every three seconds. Also note that **`top`** gives you the process identification (PID) of your running programs.

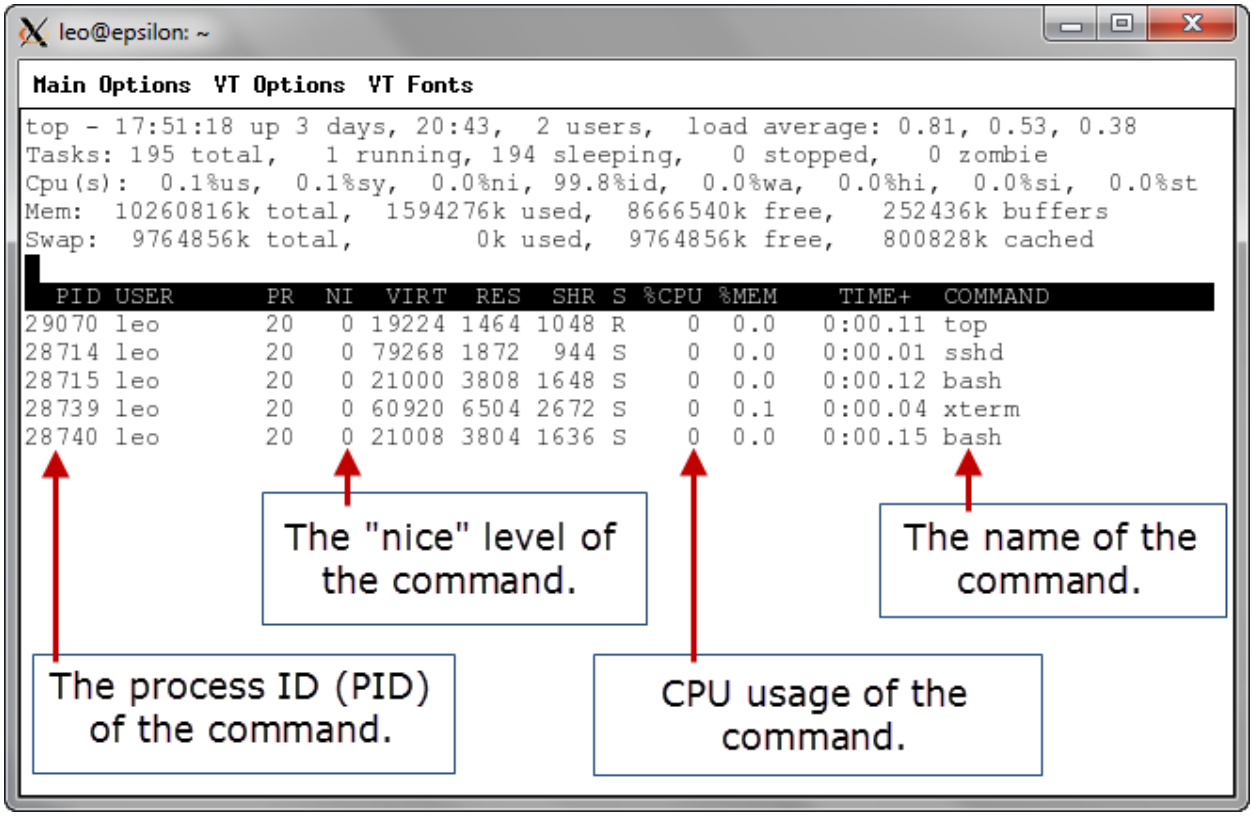
leo@epsilon: ~
Main Options VT Options VT Fonts
top - 17:51:18 up 3 days, 20:43, 2 users, load average: 0.81, 0.53, 0.38
Tasks: 195 total, 1 running, 194 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.1%us, 0.1%sy, 0.0%ni, 99.8%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 10260816k total, 1594276k used, 8666540k free, 252436k buffers
Swap: 9764856k total, 0k used, 9764856k free, 800828k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
29070 leo 20 0 19224 1464 1048 R 0 0.0 0:00.11 top
28714 leo 20 0 79268 1872 944 S 0 0.0 0:00.01 sshd
28715 leo 20 0 21000 3808 1648 S 0 0.0 0:00.12 bash
28739 leo 20 0 60920 6504 2672 S 0 0.1 0:00.04 xterm
28740 leo 20 0 21008 3804 1636 S 0 0.0 0:00.15 bash
The "nice" level of the command.
The name of the command.
The process ID (PID) of the command.
CPU usage of the command.

Figure 7: The output from top and some of the more important information it gives you.

4. In the second xterm type the command **`./myExecutable &`** and press Enter.

   Result: The second xterm will hang while the myExecutable file executes. In the original xterm where you have **`top`** running you will see **`myExecutable`** appear for about 30 seconds and then disappear when it has finished.

5. In the second xterm press Enter. Type the command **`./myExecutable &`** and press Enter. Type the command **`exit`** and press Enter.

   Result: By entering the **`exit`** command in the second xterm, you have essentially logged out. The window will close. However, in the original xterm where you have **`top`** running you will notice that the xterm process has disappeared but that the **`myExecutable`** process is still running.

6. Click on the original xterm window where you have **`top`** running in order to make the window active. Press the **`q`** key on your keyboard to end the **`top`** program (**`q`** stands for **q**uit). You should get an output similar to the following:

   ```
   [1]+  Done                    xterm
   ```

   Additional information: The output tells you that the xterm command has finished but **there is no information about the `myExecutable` command that you were running in the background or any output from it**. (Recall that **`myExecutable`** outputs a line of text telling you that it has finished. You did not see this line displayed because you closed the second xterm.)

What you have seen is that a command you run in the background will continue to run after you log out. However, any output from the program will be lost (you did not see the `myExecutable has finished` message after you logged out even though **myExtectuable** continued to run). This is because by default output messages are written to the xterm that the program was started in. When you log out the xterm is closed so there is no place to output messages from your program. It is crucial that you get these messages if something goes wrong during the execution of a background process, but if the xterm is closed (which is what happens when you log out) those messages are lost.

In order to get these output messages you can use the **nohup** command (**nohup** is short for **no h**ang **up**). The **nohup** command lets a program continue running in the background after you log out. It creates a file called nohup.out where all of the output from the program running in the background will be stored so you can look at it later. You will now repeat what you did in the previous steps, except that this time you will use **nohup** on the background program.

7. Type the command **xterm &** at the command prompt and press Enter. In the same xterm, type the command **top –u** *yourUserName* and press Enter.

   Result: A second xterm window appears. In the original xterm window you should get a display similar to Figure 7.

8. In the second xterm type the command **nohup ./myExectuable &** and press Enter. You get an output similar to:

   ```
   nohup: ignoring input and appending output to `nohup.out'
   ```

9. Press Enter. Type **exit** and press Enter to close the new xterm.

   Result: Again, the second xterm closes, but you will see that **myExecutable** continues to run in the **top** output in the original xterm. Wait until **myExecutable** has finished (until it disappears from the list of commands in the original xterm).

10. In the original xterm press the letter **q** to end the top program. Type **ls –l** in the xterm and press Enter.

    Result: There is a file called `nohup.out` in the listing of files. This file contains all the output that was generated from **myExecutable**. You will verify this next by using the command **less** to read the contents of the file.

    Additional information: The command **less** *filename* outputs the contents of the file *filename* one screen at a time so you can read it. You can use the arrow key to scroll up and down. Press the spacebar to advance the file one screen. You can search the file for a term by pressing the '**/**' key and entering the term you want to search for. Use **less** rather than **pico** to read a file because there is no chance of altering the file with **less**. Press the letter **q** on the keyboard to exit from **less**.

11. Type **less nohup.out** in the xterm and press Enter. The output should look something like:

```
myExecutable has finished
nohup.out (END)
```

This verifies that the output from **myExecutable** was stored in `nohup.out`. Type the letter **q** on the keyboard to exit from **less**.

This tutorial has shown you that running a program in the background allows the program to continue running after you log out, but you will not see any output messages from the program after you log out. Use the **nohup** command so that output and error messages are stored in a file called `nohup.out`. The command **less** will allow you to see the contents of the `nohup.out` file.

Quick Quiz:

You should be able to answer the following questions about **top** and **nohup**:

A. How can you see all of the processes you are currently running?
B. What is the advantage of starting a background process with **nohup**?
C. How do you read the contents of your `nohup.out` file?
D. How do you verify that your background process has finished?

5.5 Ending a process with the `kill -9` command

There are occasions when you may want to end a program that is misbehaving. An example of a misbehaving program is one in which you have an infinite loop. Another example would be a long simulation that was given the wrong input. In both of these cases you will want to stop the program from running. To do this you use the **kill -9** command.

1. Log on to the RAMLab (see section 2 *How to Log on to the RAMLab Locally* or 4 *Logging on to the RAMLab Remotely*).

2. Type the command **xterm &** at the command prompt and press Enter.

   Additional information: To end the process you need to know its PID. You can get the PID by using the **top** command.

3. Type **top -u** *yourUserName* in the original xterm and press Enter. Find the *PID* of the xterm process you just started.

4. Type **kill -9** *PID* in the xterm (where *PID* is the value you found in step 3.) and press Enter.

   Result: The second xterm closes. If you run **top -u** *yourUserName* you will see that the **xterm** process and the *PID* are not listed in your running programs (that is, the xterm program has been killed).

Quick Quiz:

You should be able to answer the following questions about stopping processes in Linux:

A. How do you stop a program from running?
B. What is the most important thing you need to know in order to kill a process?
C. How can you find the PID number?

6 Common Programs

This section goes over some of the programs commonly used in the RAMLab. These programs include MATLAB, Intel Fortran (or ifort), and the gcc c compiler.

6.1 MATLAB

MATLAB is a popular program for performing computational work in applied mathematics. To start the graphical integrated development environment enter the following command in an xterm:

matlab &

If you are running a long simulation under MATLAB then you will not want to start the program in its graphical mode. In this case you will want to start the program in the background using the following command:

nohup matlab –nodisplay < *yourMATLABFile.m* **>** *yourOutputFile.txt* **&**

- The **matlab –nodisplay** command starts MATLAB in text mode.
- The **<** *yourMATLABFile.m* tells MATLAB to use *yourMATLABFile.m* as its input.
- The **>** *yourOutputFile.txt* tells MATLAB to send its output to the file *yourOutputFile.txt*.

You should already be familiar with the use of **nohup** and **&** in Linux. If not, see sections 5.3 *Foreground and background processes* and **5.4** Using top and nohup with background processes in this user guide.

You can get MATLAB to email you when your program has finished running. To do so enter the following line at the bottom of your MATLAB file:

!pwd | mail –s finished *yourEmailAddress*

6.2 Intel FORTRAN (ifort)

The FORTRAN compiler installed on the RAMLab workstations is Intel's **ifort** compiler. The following is a short tutorial on using **ifort** to write a "Hello world" program.

1. Log on to the RAMLab (see section 2 *How to Log on to the RAMLab Locally* or 4 *Logging on to the RAMLab Remotely*).
2. Type **pico hello.f** in the xterm and press Enter.
3. Type the following lines of text in the pico window (start each line with a tab and press Enter at the end of each line):

```
write(unit=*,fmt=*) 'Hello world from ifort'
end
```

4. Press ctrl-o to save the file and ctrl-x to quit pico.
5. Type **ifort hello.f –o hello** in the xterm and press Enter.

   Additional information: **ifort** takes a FORTRAN source code file with a .f extension as input. The **–o hello** part of the command tells **ifort** to put the

output file (the executable) in a file called **hello**. If you do not give the –o option then **ifort** will place the executable in a file called `a.out`.

6. Type **ls –l** in the xterm and press Enter.

   Result: The file `hello` should be listed in your directory. You should have read, write, and execute permissions on this file.

7. Type the command **./hello** in the xterm and press Enter. You should get the following output:

   ```
   Hello world from ifort.
   ```

6.3 The gcc c compiler

The c compiler installed on the RAMLab workstations is called **gcc**. The following is a short tutorial on using **gcc** to write a "Hello world" program.

1. Log on to the RAMLab (see section 2 *How to Log on to the RAMLab Locally* or *4 Logging on to the RAMLab Remotely*).
2. Type **pico hello.c** in the xterm and press Enter.
3. Type the following lines of text in the pico window (press Enter at the end of each line):

   ```
   #include <stdio.h>
   void main(int argc,char *argv[]){

   printf("hello world from gcc\n");


   }
   ```

4. Press ctrl-o to save the file and ctrl-x to quit pico.
5. Type **gcc hello.c –o hello** in the xterm and press Enter.

   Additional information: **gcc** takes a c source code file as input. The **–o hello** part of the command tells **gcc** to put the output file (the executable) in a file called **hello**. If you do not give the –o option then **gcc** will place the executable in a file called `a.out`.

6. Type ls –l in the xterm and press Enter.

   Result: The file `hello` should be listed in your directory. You should have read, write, and execute permissions on this file.

7. Type the command **./hello** in the xterm and press Enter. You should get the following output:

   ```
   hello world from gcc
   ```

7 RAMLab Etiquette

It is important to the Department of Mathematics that the RAMLab be an open and collegial environment for all users. The following should be used as a guideline for what is considered acceptable and unacceptable behaviour in the lab.

- Treat all users of the RAMLab with respect.
- Offensive materials are not permitted in the RAMLab (or in Ryerson University in general).
- The use of the RAMLab workstations for non-academic work is **not** permitted.
- The RAMLab is an open lab. You cannot reserve a machine if you are not there to use it.

8 Quick Reference Guide

To…:	Command:	Usage Examples:
change to another directory	**cd**	**cd** *path/to/new/directory* **cd** **cd ..**
make a directory	**mkdir**	**mkdir** *newDirectory*
print current working directory	**pwd**	**pwd**
remove/delete a file permanently	**rm**	**rm** *fileName*
copy files from one location to another	**cp**	**cp** *originalFile fileCopy* **cp** *originalFile path/to/directory/fileCopy*
move a file from one location to another	**mv**	**mv** *originalFile newFile* **mv** *originalFile path/to/directory/newFile*
make a file executable	**chmod**	**chmod u+x** *fileName*
list the contents of a directory	**ls**	**ls** **ls –l** **ls -al**
create a directory (make a directory)	**mkdir**	**mkdir** *newDirectory*
run a process in the background	**&**	*executableFile* **&**
create a file using pico	**pico**	**pico** *fileName*
show your current running processes	**top**	**top –u** *yourUserName*
open a second xterm window	**xterm**	**xterm &**
log out of the system	**exit**	**exit**
read the contents of a file	**less**	**less** *fileName*
stop a program from running	**kill -9**	**kill -9** *PID*